Problem facing

Problem facing

1. 因為個lidar sensor, 唔可以直接stream 個畫面出嚟, 所以應該需要一條code stream 個screen 出嚟（例如螢幕右上角, size 就要再confirm)

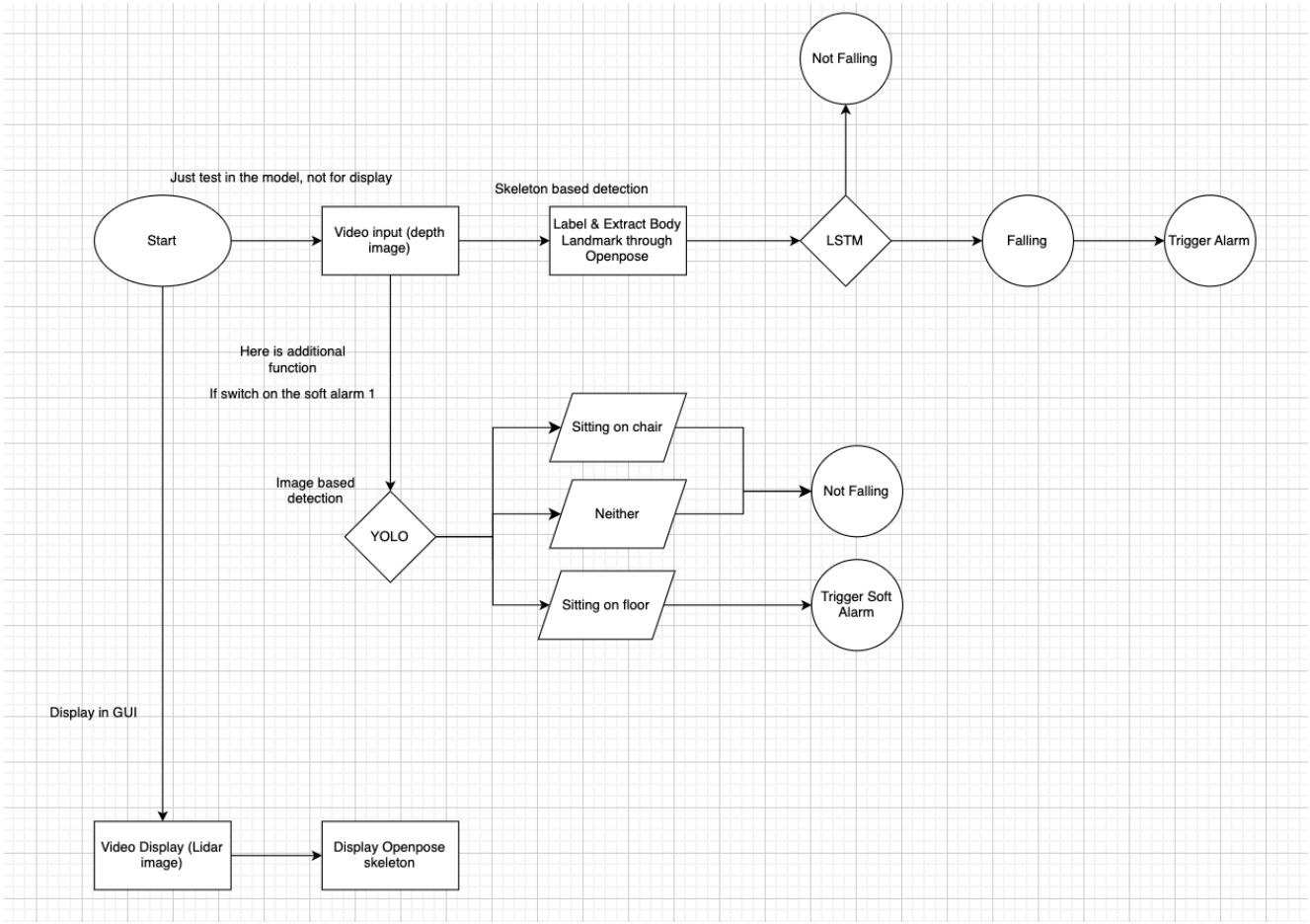
2. 係LSTM +openpose方面, 個model 基本上算係砌好。(參考 https://github.com/YJZFlora/Fall_Detection_Deep_Learning_Model)但係因為d training dataset 係本身人哋GitHub 上面已經go through 左openpose 嘅json file 再已經轉成左csv 檔嘅樣, 所以當想加自己嘅dataset 時, 係呢方面遇到問題。試過將條片go through openpose, 但係出嚟嘅json file,係每一幀一個file, 而唔係一個file 包含曬全條片。個思路係Video -> Images -> OpenPose skeletal point localization on human -> saved as json file -> converted into csv file -> ready to go for LSTM model for fall prediction

3. 另外因為想嘗試整到real time, 但係未話好有方向, 暫時唯一嘅參考（***Forum:*** *Is it possible to output JSON data from the OpenPose library in real-time?*https://stackoverflow.com/questions/57061757/is-it-possible-to-output-json-data-from-the-openpose-library-in-real-time ）

4. Soft alarm 方面, 打算做得比較獨立少少, 而且可以選擇on/off。其中一個soft alarm 打算用yolo v11認posture（image based), 例如認human sitting on chair, human sitting on floor, neither（呢part 嘅yolo已經train 好左）。呢個soft alarm 功能, 係打算係係特定環境先用, 例如病房, 當human sitting on floor, 係病房算係比較唔合理嘅行為, 所以soft alarm 就會響。

5. 然後係gui 方面, 呢part 暫時未有太多研究, 所以暫時嘅諗法比較複雜。為左方便train 同test, 會用個"lidar sensor" capture 出嚟嘅Depth images. 但係呢個raw 嘅depth images要go through LSTM for skeleton based 嘅human fall detection（path1), 當falling 嘅時候, 個red alarm will be triggered。然後同時if switch on the soft alarm, 個raw 嘅depth images 要go through yolo for image based 嘅detection (path2), if sitting on floor, soft alarm will be triggered. 再另外為左保障私隱（題目要求）, 只可以display個lidar sensor capture 出嚟嘅Lidar output (path3), 同時想係呢個output 加個openpose 嘅display (參考https://www.youtube.com/watch?v=9jQGsUidKHs）(PS: 所以實際上要stream 到2個independent 嘅screen 出嚟, depth image and lidar image)

New block diagram


Not Falling
Just test in the model, not for display
Skeleton based detection
Start
Video input (depth image)
Label & Extract Body Landmark through Openpose
LSTM
Falling
Trigger Alarm
Here is additional function
If switch on the soft alarm 1
Image based detection
YOLO
Sitting on chair
Neither
Sitting on floor
Not Falling
Trigger Soft Alarm
Display in GUI
Video Display (Lidar image)
Display Openpose skeleton

Intermediate Report (Just for reference)

ABSTRACT

With the aging population in Hong Kong, falls have emerged as a significant health threat, responsible for substantial morbidity and mortality among seniors, with significant implications for healthcare systems and quality of life among the elderly. This project focuses on designing and developing an artificial intelligence-based LiDAR sensor system that is non-invasive and accurate for real-time fall detection for the elderly, particularly in hospital settings. Enhancing safety in hospital environments by issuing timely alarms when falls occur, can minimize medical resource waste and improve patient outcomes. This project introduces the collection and preparation of datasets for normal activities and fall events, model development using YOLO and LSTM for object and human detection, and extensive testing of the system's effectiveness.

We conducted a thorough literature review, comparing different types of LiDAR sensors, ultimately favoring 3D LiDAR for not only its non-invasive nature and superior depth perception compared to traditional RGB cameras but also their cost-effectiveness, ability to produce 3D point cloud data, and high accuracy. By comparing current human detection models, we can find the fitting algorithm for improving the detection outcomes to identify human pose and action for human fall detection and use the human skeleton keypoint recognition method. Compared to conventional methods, which often rely on intrusive surveillance or wearable devices, the combination of AI and LiDAR technology significantly enhances the ability to detect falls. Additionally, the project addressed several limitations of current human fall detection models, such as the confusion between lying down, sleeping, and falling motions, ultimately leading to improved accuracy and reliability in fall detection.

The methodology section details the hardware setup utilizing the XT-S240 Mini LiDAR sensor, the data collection from various sources, including the UR Fall Detection Dataset, Multiple Camera Dataset, and YouTube dataset, data processing, landmark extraction, and the development of two primary models, YOLO and LSTM. We created a custom dataset specifically tailored to hospital settings, which involved capturing and annotating images of essential objects such as beds and chairs for the YOLO model. Furthermore, we integrated human pose estimation through

OpenPose for body landmark extraction and applied LSTM networks to analyze temporal movement patterns indicative of falls, thus enhancing the system's ability to recognize dynamic fall scenarios accurately.

To summarize the progress made to date, we developed a comprehensive dataset, which were annotated for training purposes. We developed two main algorithms: YOLO for object detection in complex environments and LSTM for analyzing temporal sequences of body movements to identify falls. Also, this project includes hardware functionality tests, LSTM model parameter optimization, and the challenges faced in preparing suitable datasets for YOLO training.

In future directions, we propose further enhancements on the system's capabilities, a user-friendly graphical user interface, two operational modes to adapt the system to various environments, and exploring innovative soft alarm solutions such as voice activation features for alarm triggering. Moreover, the detection algorithms must be optimized further to ensure rapid responses in critical situations and thus the system's operational readiness in real-world applications.

In conclusion, this report highlights the importance of developing innovative solutions for fall detection and combining advanced sensor technology with machine learning to tackle a critical health issue affecting the elderly population. The innovations introduced in this project will illustrate a step forward in fall detection technology, promising to contribute positively to the well-being of vulnerable populations.

Table of Contents

List of Tables

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Along with the advancement of medical technologies and society leading to better healthcare, education, and income, higher life expectancy with averages of 82.5 years for males and 88.1 years for females in 2023 remained among the highest in the world based on the Department of Health [1]. As the population aging problem becomes increasingly serious in Hong Kong, falls have become one of the major threats to the health of the elderly, being the second leading cause of accidental injury death worldwide. According to the World Health Organization (WHO), an estimated 684,000 people die from falls globally every year [6], and many falls may also cause long-term physical injuries. Therefore, the development of an efficient fall detection system is important for medical and social significance. This project aims to design and develop an artificial intelligence-based lidar sensor system to detect falls for the elderly in real-time and issue an alarm in time, thereby improving safety at the hospital.

1.1 BACKGROUND TO THE PROJECT

Falls are a significant health concern among the elderly, particularly over the age of 60. According to the WHO, they place a substantial burden on the healthcare system, with approximately 37.3 million requiring medical assistance annually [6]. Falls can be attributed to intrinsic variables like muscle strength and balance, as well as extrinsic factors such as slippery floors and poor lighting [5], resulting in functional impairments, poor quality of life, reduced abilities [7], and increased mental health issues. Half of those who suffer a long lie following a fall die within six months [4]. The aging population exacerbates this issue, straining healthcare systems due to the high costs of hospitalizations.

Conventional fall detection methods, which often rely on wearable devices or video surveillance systems, can be invasive and inefficient. These approaches face challenges such as constrained usage duration and capture space [2], which might lead to privacy concerns and a high rate of false alarms. Nonetheless, the integration of AI and LiDAR technology presents a promising, non-invasive solution for enhancing fall detection accuracy while

respecting user privacy. Developing such systems is crucial for improving health, safety, and overall quality of life for vulnerable elderly individuals.

1.2 MOTIVATION

Falls are unpredictable and sudden, making accurate fall detection complicated. Because of the aging population and the rising frequency of falls, emphasis on fall detection is vital, presenting a major health concern that needs urgent attention. Besides physical injuries, these incidents may also contribute to psychological repercussions, including loneliness and anxiety. With advancements in artificial intelligence and sensor technology, safety and quality of life improvement for the elderly can be achieved, which is the main driving force behind our research. This can lead to more effective healthcare systems by reducing medical burden, as prompt detection can facilitate quicker medical responses, which can minimize the waste of medical resources such as hospital stays and healthcare expenses spent on treating patients with fall-related injuries. We have also been inspired to develop this system by our personal experience or understanding of the troubles faced by those around us who are at risk of falling, with the hopes of providing practical assistance.

A fall can occur within a second, typically taking 0.45 to 0.85 seconds, and during a fall, a person's posture and shape change, which are important when detecting falls [3][4]. Thus, LiDAR sensors are particularly advantageous because of their high precision and heightened sensitivity to environmental changes, making them ideal and applicable for accurate fall detection. Combining AI with LiDAR technology enhances the ability to analyze data and increases the accuracy of detection systems. This focus also fosters research and development for innovative solutions that enhance the safety and welfare of vulnerable populations. Therefore, it is crucial to build an effective fall detection system to minimize the risks associated with falls and improve the quality of life for elderly individuals.

1.3 AIMS & OBJECTIVES

This project aims to design and develop an artificial intelligence-based LiDAR sensor system to monitor and detect falls accurately and efficiently for the elderly in real-time and issue an alarm in time, thereby improving safety and quality of life for patients at the hospital. The system will utilize LiDAR sensors to capture data on a change in body posture and shape of the person during the fall, employing pose estimation along with human detection algorithms and a fall detection model.

To support this, we plan to establish a dataset to gather data for normal activities and fall events, followed by thorough annotation for training purposes. The collected data will be used to train a deep learning model, to enhance fall detection accuracy. The LiDAR sensor will be integrated with the trained model into a complete system to conduct extensive testing to validate its effectiveness and reliability. We will also explore potential strategies to improve the efficiency of fall detection systems, employing a multifaceted approach tailored to specific application scenarios.

1.4 SCOPE

The scope of this project encompasses the design and development of an AI LiDAR sensor system specifically aimed at fall detection among patients, particularly the elderly. The project will be structured around several key components:

1. Dataset for Training Purposes

   The initial phase will involve the collection and preparation of a comprehensive dataset for both normal activities and fall events, as well as bed and chair for object detection. This dataset will be processed and normalized to facilitate effective training of the model.

2. Model Development: YOLO and LSTM model

   We will focus on developing two primary models: the YOLO model for object detection and the LSTM model with OpenPose for human detection and fall

detection. This dual approach will enhance the system's ability to accurately detect falls in real-time.

3. Model Training

   After the models are developed, we will conduct extensive training using the prepared dataset. This phase will optimize the models to improve detection accuracy, sensitivity, and overall performance.

For future direction, the project will also cover:

1. System Integration

   We will implement two operational modes within the system to accommodate various environments, such as a hospital ward and stairwell. This functionality will allow users to switch between modes based on their specific needs.

2. Validation

   The effectiveness of the trained model will be validated through rigorous testing with the gathered LiDAR sensor data, ensuring that the system meets the required performance standards in real-world scenarios.

3. User interface

   A user-friendly graphical user interface (GUI) will be developed, allowing caregivers to use the system easily. The interface will include features for mode switching, alarm location display, and soft alarm notifications.

4. Innovation

   The project will explore innovative solutions, such as different types of soft alarms and voice activation features, to enhance user experience and system reliability. These innovations aim to improve responsiveness and ensure timely assistance during fall incidents.

5. Demonstration: complete system with real-time

   Finally, the complete system will be demonstrated in a real-time setting, showcasing its capabilities in fall detection and alarm response. This demonstration will serve to

validate the system's functionality and effectiveness in enhancing the safety of elderly individuals.

CHAPTER 2: LITERATURE REVIEW

Sensors and detection algorithms are essential parts of human fall detection, therefore the details and consideration of sensors, LiDAR (Light Detection and Ranging) sensors in different dimensions and human motion detection algorithms are compared and explained.

To highlight how LiDAR sensors overcome the challenges in capturing data of current cameras, the working principle, advantages and limitations between the RGB cameras and LiDAR are compared. Different types of LiDAR sensors are also compared in order to discover the appropriate LiDAR sensors for fall detection.

Various human detection algorithms have been studied and compared to find the fitting algorithm to introduce in the machine learning algorithms for further processing.

A review of the current human fall detection models and their major limitations are analyzed. This provides innovative ideas and features of our systems on improving the detection outcomes.

2.1 TYPES OF CAMERAS AND SENSORS USED IN HUMAN DETECTION

2.1.1 RGB Camera

RGB cameras capture the ambient light in red, green, and blue color models. It then combines this visible light and transfers it to electrical signals in image and video format. The RGB color images form in a color filter array (CFA) in Bayer pattern arrangement [8]. In the application of human detection, they capture the human body, and clothing, and reproduce images and videos which are the same as the human eye perceives. Commercial RGB sensors, typically webcams, such as the Sony PS Eye RGB camera, are proven in human pose recognition and motion recording. They are frequently used in research and commercial applications in human detection and recognition because of their simple design and low cost [9, 10].

However, the use of RGB cameras in human detection encounters a few challenges. The major concerns of RGB cameras are safety concerns and privacy invasion if they are installed in public areas, e.g. hospitals [11]. The identifiable features of individuals, e.g. faces, clothing and other personal attributes can be captured. This raises legal and ethical concerns that users' images can be stored on the server and accessed by the company operators [9]. The use of RGB cameras in housing and public areas requires careful handling and storage of sensitive visual data. Moreover, the performance of RGB cameras may be limited under low and high ambient lighting conditions due to the highly sensitive properties to lighting conditions of RGB cameras [11, 12]. This may lead to incorrect evaluation for human detection use.

2.1.2 LiDAR Sensor

The use of LiDAR (Light Detection and Ranging) sensor technology in human detection has been increasingly high in recent years. LiDAR sensor is a device that involves optical remote sensing technology, which applies the principle of time of flight (ToF) to measure the distance between the target objects. Firstly, the LiDAR sensor emits ultrashort laser light pulses to map the target object. The laser light pulses reach the target and reflect. The photosensitive sensor detects the reflected light and calculates the flight time from the emitted light to reflect off the objects and return to the device. Consequently, the distance between the specific objects and the LiDAR sensors can be measured, based on the following equations [13-15],

$$d = \frac{c \cdot t}{2}$$

where d is the distance to the object, c is the speed of light (equal to $3*10^8$) and t is the time taken for the light to travel to the object and reflect back.

Compared to the RGB camera, the primary benefit is that the LiDAR sensor generates point cloud information, but not in RGB images. The LiDAR sensor provides limited data, for instance the human visual appearance, which is not shown in the data. This main merit avoids privacy concerns from the users [16]. Using LiDAR sensors provides depth

measurements, which can be used in the 3D mapping of the environment for identifying falls.

2.2 LIDAR SENSORS COMPARISON

	2D LiDAR	3D LiDAR	4D LiDAR (Doppler LiDAR)
Dimension	2, measures x, y axes	3, measures x, y, z axes	4, measures x, y, z axes and velocity (Doppler effect)
Data Output	2D Point Cloud	3D Point Cloud	3D Point Cloud with velocity data
No. of Light Beams Emitted	Single	Multiple	Multiple
Range	Up to 100 m	Up to 30 m	Up to 30 m
Accuracy	±10 mm ~ ±100 mm	±50 mm ~ ±100 mm	±50mm ~ ±100mm
Cost	Lowest	Moderate	Highest
Applications	Basic Distance Measurements, Person Tracking	Object Detection, Person Tracking, Autonomous Navigation	Autonomous driving, Advanced driver-assistance systems (ADAS)
Example	Sick - TiM 2D LiDAR Sensors Series	Intel® - RealSense™ LiDAR Camera L515 Toffuture - XT-S240 Mini from Toffuture	Aeva - Atlas™, Aeries™ Unitree - 4D LiDAR L1

Table 1: Comparison of different dimensions of LiDAR sensors [11, 17-19]

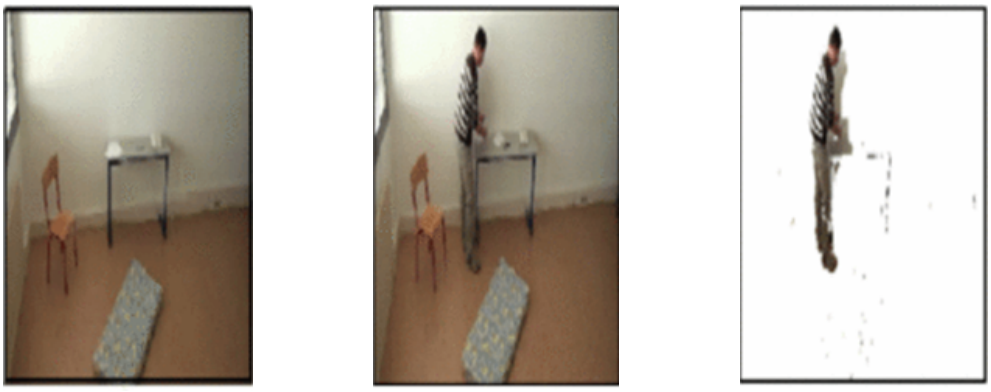
By comparing the 2D LiDAR sensor with 3D and 4D LiDAR sensors, the latter provides three-dimensional point cloud data, which enhances the depth perception of human fall motion captured [11]. It allows understanding of video data in comprehensive and dynamic environment applications, such as rooms, and backstairs. Moreover, the 3D and 4D LiDAR sensors provide higher accuracy and resolution of image and video data. In our research topic, we currently focus on the spatial positioning of human fall detection. The use of 3D LiDAR sensors fulfils the data requirements and thus we apply the 3D LiDAR in this research.

2.3 Human Detection Algorithms

2.3.1 Body shape-based Method

Current vision-based approaches mainly use the continuous video surveillance of cameras to detect fall motion based on various classification methods [20], for example using the human body shape-based or human skeleton keypoint estimation.

In the human body shape-based method, collected video images in frames are transmitted for image preprocessing and segmentation. To extract the human body image from the background pixels, background subtraction algorithm is typically used as the segmentation method [21, 22]. Background model is built by using successive frame differences for labelling stationary pixels. The algorithm lists out a sequence of consecutive images and considers the corresponding pixel as background if the selected pixel remains stationary for N selected numbers of frames continuously. According to Figure 1, the difference between the value of image acquired I(x,y) and the background image B(x,y) is computed and the human body shape can be processed without background pixels [21-23].



(a) (b) (c)

Figure 1: Extraction of human body using background subtraction algorithm: (a) Background image, (b) Human image with background, (c) Processed human body image [21]

After the background subtraction algorithm, human shape image is further processed into a contour image in binary by edge detection. Several researches applied the shape model for human motion tracking. [24-26] proposed using human surveillance shape matching for tracking and analyzing human shape deformation to capture motion dynamics and even human fall detection. The algorithm behind this shape-based method is simple comparatively, because of the computation of the difference between the image model and background model. One major disadvantage is difficulty in distinguishing a moving human from other moving objects, especially for multiple persons moving and overlapping. The confusion in identifying the background objects and human body in a dark environment raises another challenge [22]. Also, it cannot provide detailed internal joint information in classifying similar poses, for instance lying down and falling [20]. This can lead to misinterpretation of the body posture and movements for system processing and computing, leading to inaccurate classification results.

2.3.2 Skeleton Keypoint Recognition Method

Human skeleton keypoint recognition method focuses on the association of body parts using nodes, forming the skeleton information of each person. Various open-source projects, for instance, OpenPose [27, 28] and OpenPifPaf [29], which are commonly used in human detection interpret the human skeleton for image processing. The method follows the human body's top-down approach and pinpoints the more than hundreds of nodes, e.g. knees, hips, and shoulders [27-30]. Therefore, these projects can identify full body, hand, foot movement and facial expression.

Figure 2: Human skeleton keypoint recognition (OpenPose) [28]

By using the model and feature fields mapping within these projects, e.g. part affinity fields for OpenPose and part intensity and association field for OpenPifPaf, a distribution and location map of human feature nodes is processed [27-29]. This ensures the framework to identify and join the nodes with the corresponding individual, even in a crowded environment where multiple people overlap. This method is able to estimate multiple persons' poses and avoid misinterpretation brought by overlapping. 3D Pose detection is also available in this method. Once the 2D nodes are identified, they can be triangulated based on their semantic understanding of detected joints, enhancing them to assemble the 3D pose image [27-29]. Since the 3D pose estimation provides detailed depth information, it allows better detection and differentiation of various body orientations and movements for further analysis and application, which enhances the accuracy of the analysis model.

2.4 Limitations of Current Human Detection Model

We reviewed various existing proposed machine learning models of human fall detection found on GitHub and Roboflow. They all demonstrated the basic detection of human fall motion in images and video format. However, these models [31, 32] still pose some limitations, mainly the difficulties in defining the motion of lying down and sleeping from fall detection, as these postures are similar and share common features in machine learning algorithms. This would lead to an increase in false positive and false negative cases, thus decreasing the accuracy and precision of the human falling detection model. Therefore, a solution must be developed in order to overcome the distinction between lying down, sleeping and falling motion.

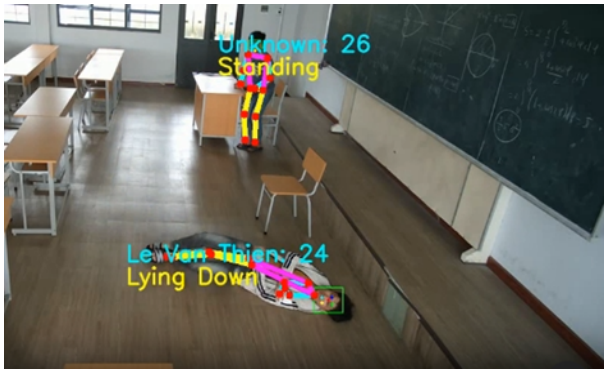


Figure 3 (Left): False positive case, as the model labels the person who is lying down as fall [31]

Figure 4 (Right): False negative case, as the model labels the person who falls as lying down [32]

2.5 Concluding Remark

Comparing the RGB cameras with the LiDAR sensors, the result proves that visual data captured from LiDAR sensors solves the privacy concern brought by the RGB cameras, as point cloud data is outputted instead of RGB images. This type of sensor can provide depth information, enhancing the mapping of the applied environment and accurate localization of human fall motion. This validates that LiDAR is the optimal choice for accurately detecting falls in different environments.

From the comparison of 2D LiDAR sensor, 3D LiDAR sensor and 4D LiDAR sensor (Doppler sensor), we understand the requirements for considering appropriate LiDAR

sensors in human fall detection. The result highlights the superiority of 3D LiDAR sensors due to their cost-effectiveness, ability to produce 3D point cloud data and high accuracy, which are able to be applied in human fall detection and other environmental object detection.

Based on the study of the working principle of human detection algorithms, we understand the importance of these advanced algorithms to identify human pose and action for human fall detection. The study features the benefit of using the human skeleton keypoint recognition method. Understanding the strengths and limitations of the human skeleton keypoint recognition method enhances our further selection in the building of the machine-learning model for accurate fall detection.

By comparing the limitations of current human detection models, we understand the main constraint is the confusion between lying down, sleeping and falling motion. This provides us with directions on developing an additional model to group the normal daily activities from falling motion to increase the accuracy and precision of our designed model.

CHAPTER 3: METHODOLOGY

3.1 HARDWARE

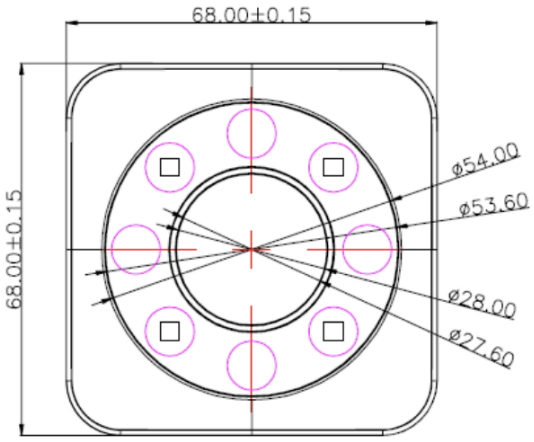


Figure 5 (Left): XT-S240 Mini sensor [15]

Figure 6 (Right): XT-S240 Mini sensor dimension [15]

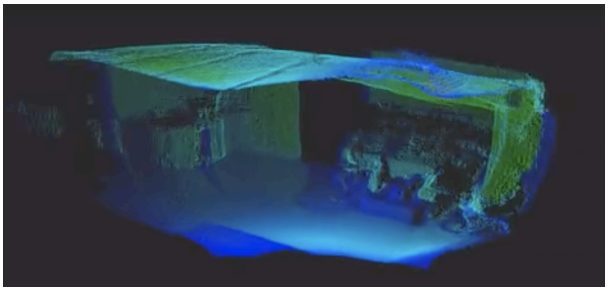


Figure 7: Comparison of raw image captured by camera (Left) and XT-S240 (Right)

Our LiDAR-based fall-detection setup uses XT-S240 Mini to detect the environment. XT-S240 Mini is a pure solid flash ToF LiDAR with a big FOV of 106° X 80°, high resolution equivalent to 240 lines supporting 76,800 pixels/frame, and a frame rate of 1-20 frames per second. For its indoor detecting range, it can reach 12m [15]. XT-240 Mini can

support real-time monitoring, typically used in volume measurement, access and security, and body counting. A key consideration when configuring this sensor is based on its cost-effectiveness. Due to its mature silicon-based semiconductor process, reliability, consistency, and stability are guaranteed even with a comparatively low cost.

Another fascinating feature of this LiDAR sensor is it can directly output a 3D point cloud. The single emitted laser pulse generated from the LiDAR can be bounced back as multiple returns when encountering numerous surfaces along its path. Based on the detection order, a LiDAR system can distinguish between different types of returns and capture very dense points produced by laser pulses [33]. As shown in figure 7, with every point captured and rendered in real-time, the 3D point cloud displays the precise locations of objects and formulates 3D representations of the surroundings within the environment. LiDAR differs from cameras as it does not capture color information but instead records 3D distance data to form a point cloud. This process creates a privacy-protected, anonymized representation of the scene without detailed visual information.

This LiDAR sensor also outputs Depth images, a 2D representation of a scene in which each pixel encodes the distance from the camera to a point in the scene. It provides information about the spatial layout of objects in the image enabling automated object recognition and classification, allowing for the identification and categorizing of features like humans and infrastructures. Further detail will be discuss later. Moreover, a software development kit (SDK) is provided to support the processing and streaming of depth and amplitude images captured by XT-S240 Mini. It removes invalid noise, optimizes image quality, and highlights essential features within the pixel.

3.2 DATA SET

There are 3 different data sets, namely – 'UR', 'MC', 'YOUTUBE'. There are 484 videos, including 252 videos with falls and 232 without falls.

3.2.1 UR Fall Detection Dataset (UR)

Figure 8: Example of video in UR Fall Detection dataset [35]

This dataset contains 70 (30 falls + 40 activities of daily living) sequences. Fall events are recorded with two cameras (camera 0 from the side angle and camera 1 from the directly above the angle) [35]. However, the activities of daily living events are recorded with only one device (camera 0), so only the data from camera 0 is used for the project. Also, from a side angle, the image frame is easier for AI recognition. Using camera 0 can also mimic the view of a camera set from a room.

In order to match body landmark data with the labeled UR dataset, it is necessary to transform the labels. The UR dataset originally contains label files for each frame in CSV format, with each row representing a sample of data corresponding to an image frame of a video. First, the sequence name - camera name can be ignored because all of the samples used for this project are from camera 0. Second, after reviewing the original data, a new labeling system was implemented. In this system, all non-falling situations are classified as' 0' while situations where personnel had fallen down or are falling down are now classified as '1'.

3.2.2 Multiple Camera Dataset (MC)

Figure 9: Example of video in Multiple Camera Dataset [36]

This dataset contains 24 scenarios recorded with 8 points of view, in total 192 sample videos. The first 22 first scenarios contain a fall and confounding events, the last 2 ones contain only confounding events [36].

Each video may contain some personnel performing a falling action. For the accuracy of extracting body landmarks, each video is tailored to contain only one person at a time. This dataset does not contain any label file. Therefore, each video is separated into 2 parts: one only contains frames of personnel who have fallen down or are falling down, and the other only contains frames of all non-falling situations. Later, all the frames of the first part are marked to be fall(1), while all frames of the second part are marked to be not fall (0).

3.2.3 YouTube Dataset

This dataset contains a total of 23 videos recording falling action, which are all sourced from YouTube. It includes videos of teaching the elderly how to fall safely and videos of various settings for the elderly to fall. For later labeling, each video is separated into 2 parts: one only contains frames of personnel who have fallen down or are falling down, and the other only contains frames of all non-falling situations. Later, all the frames of the first part are marked to be fall(1), while all frames of the second part are marked to be not fall (0).

3.3 Data Set Processing

3.3.1 Body Landmark Extraction

OpenPose, an open-source tool, is used to produce body landmarks for frames of each video. [37] OpenPose was the first real-time multi-person system to jointly detect the human body, hand, and foot, which are key points in single images. Our system utilized the 2D real-time multi-person keypoint detection with 25-keypoint body/foot keypoint estimation.

All body landmarks for each frame are stored in a JSON file. The generated body landmarks for each frame contain 25 key points [37].

```
// Body parts mapping
const std::map<unsigned int, std::string> POSE_BODY_25_BODY_PARTS {
    {0,  "Nose"}, {1,  "Neck"},{2,  "RShoulder"}, {3,  "RElbow"}, {4,  "RWrist"},
    {5,  "LShoulder"}, {6,  "LElbow"}, {7,  "LWrist"}, {8,  "MidHip"}, {9,  "RHip"},
    {10, "RKnee"}, {11, "RAnkle"}, {12, "LHip"}, {13, "LKnee"}, {14, "LAnkle"},
    {15, "REye"}, {16, "LEye"}, {17, "REar"}, {18, "LEar"}, {19, "LBigToe"},
    {20, "LSmallToe"}, {21, "LHeel"}, {22, "RBigToe"}, {23, "RSmallToe"}, {24, "RHeel"},
    {25, "Background"}
};
```

Figure 10: Body landmarks mapping [37]

For fall detection, body landmarks of the face are unnecessary. Therefore, after reviewing the body landmarks from JSON files, 4 key points: {15, "REye"}(right eye), {16, "LEye"}(left eye), {17, "REar"}(right ear), {18, "LEar"}(left ear), are removed for each frame.

In case of an unstable connection with Google Colab or a crash with a local machine, The body landmarks are stored in CSV files, which have a higher loading speed than re-reading all JSON files.

3.3.2 Normalization

```
# Normalized the data
def normalize(list, width, height):
  for record in list:
    for i in range(0, 63, 3):
        record[i] = float(record[i])/ width
        record[i + 1] = float(record[i + 1]) / height

normalize(ur_data, 640, 240)
normalize(mc_fall_data, 720, 480)
normalize(mc_notfall_data, 720, 480)
normalize(youtube_fall_data, 640, 360)
normalize(youtube_notfall_data, 640, 360)
```

Figure 11: Code for data normalization

To normalize data according to the size of videos in different datasets, the width and height of the videos are divided as the above parameter.

```
# Count the number of "Fall" and "Not fall" in Label

fall = label.count(1)
not_fall = label.count(0)

print("Number of Fall:", fall)
print("Number of Not fall:", not_fall)
print("Number of All Data:", len(label))
```

```
Number of Fall: 13128
Number of Not fall: 22151
Number of All Data: 35279
```

Figure 12: Code for counting the number of normalized data of “fall” and “not fall”

There are 22151 frames that are not fall(0) and 13128 frames of fall(1).

3.3.3 Train-Test set Split

```
# Split data into training dataset and validation dataset
# Divided into 6:4

np.random.shuffle(all_data)
split_pt = len(all_data) // 10 * 6

train_data = all_data[:split_pt]
valid_data = all_data[split_pt:]

print("Number of Training Data:", len(train_data))
print("Number of Validation Data:", len(valid_data))

# Check if all data has been split
print((len(train_data)+len(valid_data)) == len(label))
```

```
Number of Training Data: 21162
Number of Validation Data: 14117
True
```

Figure 13: Code for train-test set split

The data is randomly shuffled to ensure that each data point creates an 'independent' effect on the model without being biased by the points that precede them. Subsequently, the dataset is split into a training set and a test set in 6:4 ratio.

3.4 PROPOSED ALGORITHMS

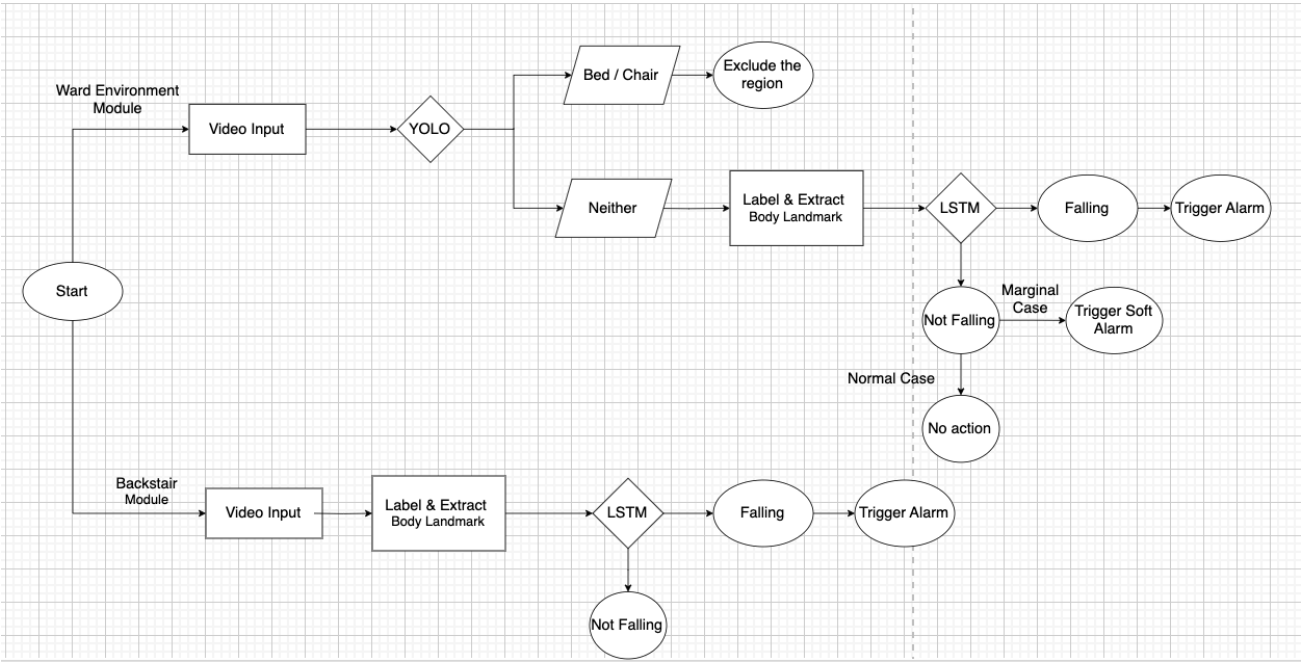


Figure 14: Overall workflow of the proposed algorithms

In our algorithms for this project, two modules are typically used to detect falls in two distinct environments: ward (room) settings and backstairs settings. The main difference between the two modules is the inclusion of YOLO object detection and some soft alarms.

The ward (room) setting is more complex than the backstairs setting. Thus, the percentage of false positives (FP) caused by daily living activities is believed to be higher than in the backstairs setting. The algorithm will perform better by omitting the daily activity chair and bed. In certain marginal cases, like someone merely bending down to pick something up or a patient trying to leave the bed, the soft alarm will be triggered as a potential alert sign for the system.

For the backstairs setting, which is a more straightforward scenario, there are not many activities of daily living, so only the falling action is concerned. Splitting two modules aims to simplify the falling detection algorithm for a simpler environment; this can eliminate

redundancy, leading to faster execution speed. This can be beneficial for applications that require real-time processing.

3.4.1 YOLO Object Detection

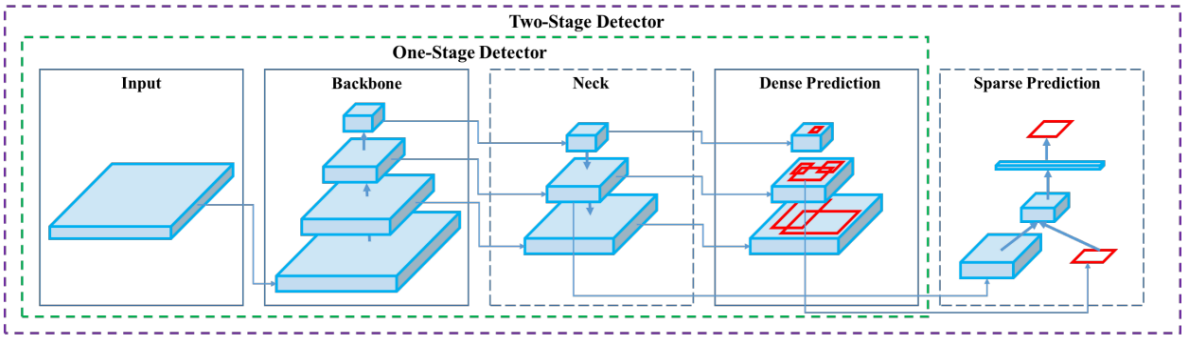


Figure 15: Two concepts of architectural object detection [37]

Object detection is critical for excluding daily activity in certain regions in our system. Rather than using other typical object detection algorithms, YOLO-V11 is used in this project. The above figure illustrates two concepts of essential mechanics of object detection models. At the core of both types of detectors, the architecture consists of three fundamental components: a backbone, neck, and head. The backbone, typically a pre-trained Convolutional Neural Network (CNN), extracts feature maps from an input image at multi-scale.

The neck then consolidates these feature maps using path aggregation blocks like the Feature Pyramid Network (FPN) to enhance feature representations across different scales before being passed to the head. The head is responsible for object classification and bounding box prediction. The head can comprise one-stage models, such as YOLO. Alternatively, it may incorporate two-stage algorithms like the R-CNN series [37,38]. The reason for choosing YOLO is because it is a one-stage model, which processes the entire image directly over a dense sampling of locations to predict the presence of objects within the environment. They are ideal for real-time object detection with limited computational resources, offering high-speed processing [39].

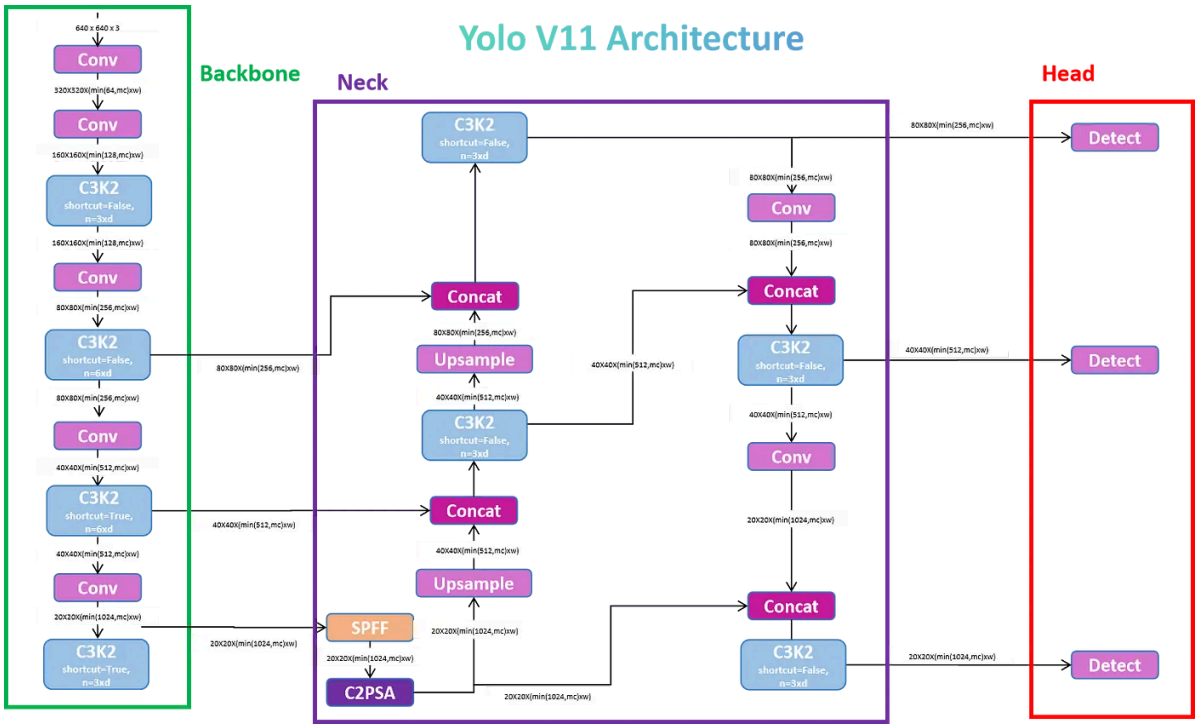


Figure 16: YOLOv11 model architecture [40]

The architecture of YOLOv11 is designed to optimize both speed and accuracy, building on the advancements introduced in earlier YOLO versions. The main architectural innovations in YOLOv11 is the presence of the C3K2 block, the SPFF module, and the C2PSA block, which enhance its ability to process spatial information while maintaining high-speed inference [41].

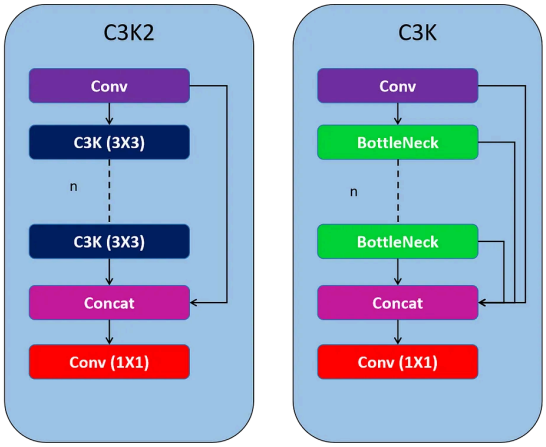


Figure 17: Structure of C3K2 and C3K blocks [40]

YOLOv11 utilizes C3K2 blocks for feature extraction, significantly enhancing efficiency. These blocks split and process feature maps using smaller 3x3 convolutions, which improves speed and reduces computational load [41]. By processing and merging these smaller feature maps, C3K2 blocks enhance feature representation.

Unlike the C2F blocks in YOLOv8, which use a single large convolution, C3K2 blocks incorporate two smaller convolutions. The "k2" in C3K2 denotes the smaller kernel size, contributing to faster processing without sacrificing performance [41,42]. By employing two smaller convolutions instead of one large convolution, computational overhead is reduced, leading to faster feature extraction.

The C3K2 uses C3K block to process the information. The C3K structure is similar to C2F but avoids splitting the feature maps, thereby enhancing data flow through a series of ‘n’ Bottleneck layers with concatenation. This design maintains a balance between speed and accuracy, leveraging the Cross Stage Partial (CSP) Bottleneck to improve overall performance [41,42]. As a more compact variant of the CSP bottleneck, C3K2 makes the architecture more efficient in terms of trainable parameters.

YOLOv11 keeps the SPFF module (Spatial Pyramid Pooling Fast) to better recognize objects of different sizes by pooling features across various image regions. By using different max-pooling operations, SPFF combines information from different resolutions, aiding in the detection of even small objects. This maintains real-time processing speeds while enhancing multi-scale object detection capabilities. The addition of C2PSA block in YOLOv11 enhances spatial attention through two PSA modules, improving focus on crucial image regions [41]. By balancing computational efficiency with detection accuracy, this block helps the model excel in scenarios requiring detailed object detection, surpassing older versions like YOLOv8.

3.4.2 OpenPose and LSTM

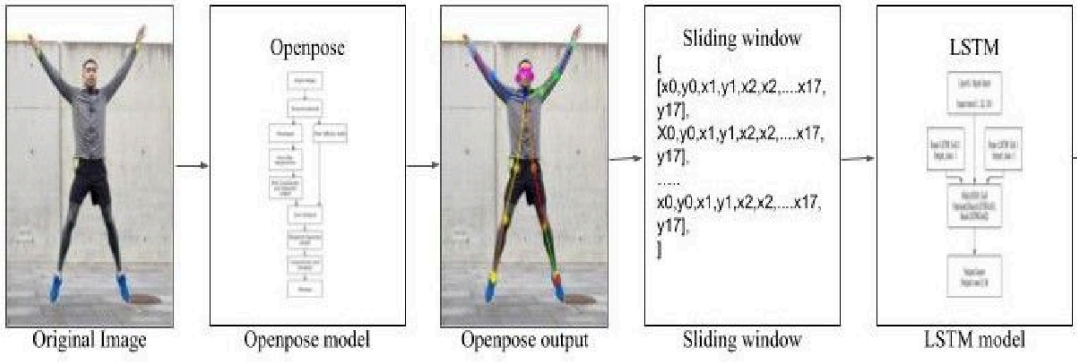


Figure 18: Overall workflow of Openpose and LSTM [43]

The input to this process consists of videos, which are analyzed as a sequence of images, processed frame by frame. Human pose key points are extracted from each frame using OpenPose, an open source for bottom-up human pose detection known for its efficiency in processing low-resolution images and lower hardware requirements compared to other state-of-the-art algorithms [43,44].

For every frame captured by each camera, a list of sets of key points is generated, with each set representing the joint positions of a person. Features are then extracted from this pose information and fed into an LSTM neural network to make initial judgments about the subject's state (falling or not falling) [43,45,46].

Subsequently, a tracking process is initiated to detect fall situations, determining whether the subject has fallen or is in the process of falling. This tracking system provides early warnings for potentially dangerous behaviors like falls.

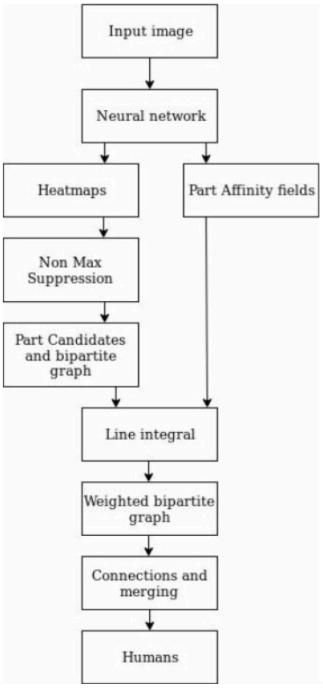


Figure 19: Openpose architecture [43]

Real-time multi-person 2D pose estimation is crucial for enabling machines to understand humans in images and videos. OpenPose effectively performs pose estimation in crowded scenes and can detect multiple people simultaneously. It identifies key points of the human body, including facial expressions, hand positions, and foot key points, allowing for accurate detection of body orientation and position [36].

In our system, the OpenPose model is trained with the COCO dataset to extract 21 body key point coordinates. OpenPose utilizes a neural network to process images, generating heatmaps and part affinity fields to pinpoint key locations and connections. Heatmaps indicate the likelihood of body parts in pixels, while part affinity fields show relationships between key points. A Non-Max Suppression layer refines these predictions by identifying local maxima. By creating a bipartite graph, OpenPose connects body parts to form skeletons, enabling detailed pose estimation for complex scenarios like crowded scenes [43,44,46].

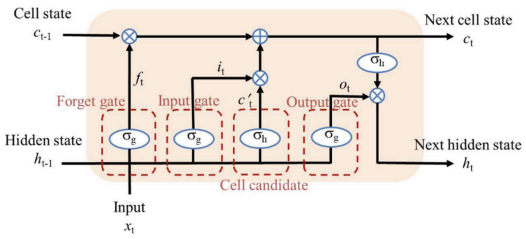


Figure 20: The typical structure of LSTM [49]

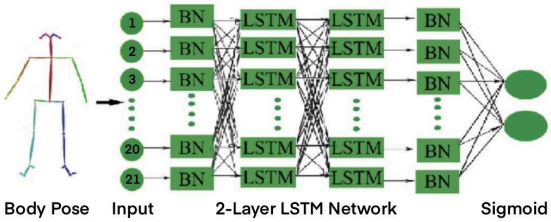


Figure 21: The network of Openpose and LSTM [47]

LSTM is a type of RNN suitable for capturing and analyzing temporal dependencies within sequential data. The gates in LSTM maintain and update a context of past movements while focusing on relevant new data [47]. It helps to learn and recognize complex movement patterns over time. As falls are dynamic events that change over time, detecting falls requires recognizing a sequence of movements and postures that indicate instability and loss of balance.

In LSTM architectures, a memory cell is regulated by three gates: the input gate, the forget gate, and the output gate. These gates determine the information to include, remove, and output from the memory cell, respectively [48,49]. This allows LSTM networks to selectively store or remove information as it progresses through the network, which allows them to learn long-term dependencies.

The forget gate in an LSTM network discards unnecessary information from the cell state. It processes the previous hidden state, h_{t-1}, and the current input, x_t, to determine what to remember and what to forget from the past cell state. While the input gate receives the previous hidden state, h_{t-1}, and the current input, x_t, produces a vector of values between 0 and 1, indicating how much of the current input should be included in the cell state. The output gate receives the previous hidden state, h_{t-1}, the current input, x_t, and the current cell state, c_t, generating a vector of values between 0 and 1 to determine the proportion of the current cell state to output as the current hidden state, h_t [48,49].

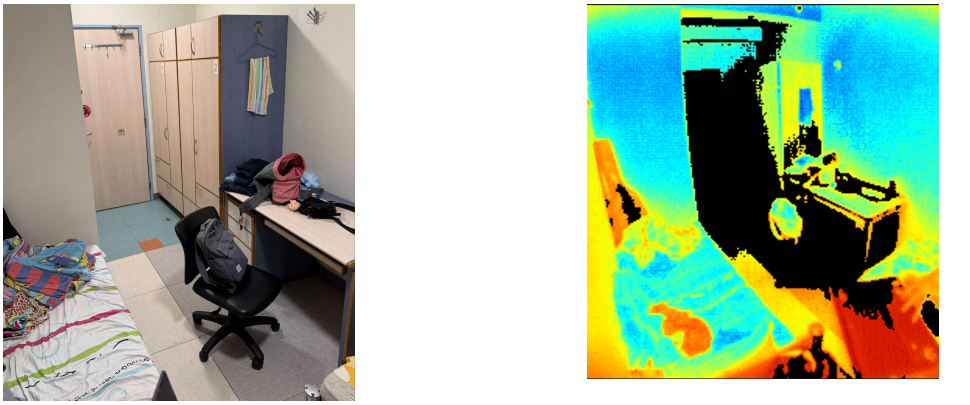
Therefore, due to the high complexity of falling movement, as it involves multiple joints and body parts, LSTM effectively handles such spatiotemporal data by processing multiple time

steps and recognizing the coordinated movement of different body parts, making them ideal for predicting falls based on the body postures changing over time [43,47,48,49].

CHAPTER 4: TASKS COMPLETED

4.1 HARDWARE FUNCTIONAL TEST

After acquiring the sensor, we conducted various tests to explore its functionality. For instance, we captured images of the environment using different modes. The following images demonstrate the results obtained in amplitude mode and depth mode.



Raw Image **XT-S240 Depth & Amplitude Image**

Figure 22: Comparison of raw image captured by mobile phone camera (Left) and XT-S240 (Right)

One of our objectives is to simulate a hospital ward setting. However, since accessing an actual ward for photo capturing was not feasible, we used a hall resident room setup as an analogous environment for our simulations.

Besides, we encountered several issues with using the SDK provided by the sensor manufacturer. Subsequently, we need to reach out to the technical support team for further assistance in utilizing the SDK to access additional functions.

4.2 LSTM Model

4.2.1 LSTM Model Parameter

```
def train_lstm_model(x_train, y_train):
  x_train = array(x_train)
  x_train = x_train.reshape((len(x_train), 1, len(x_train[0])))

  y_train = array(y_train)

  lstm_model = Sequential()
  lstm_model.add(LSTM(16,
                input_shape=(1, 63),
                return_sequences=True))
  lstm_model.add(LSTM(16, ))
  lstm_model.add(layers.Dense(1, activation='sigmoid'))
  lstm_model.compile(optimizer='rmsprop',
                loss='binary_crossentropy',
                metrics=['acc',
                        metrics.AUC(),
                        metrics.FalseNegatives(),
                        metrics.Recall(),
                        metrics.Precision(),
                        metrics.FalseNegatives(),
                        metrics.TrueNegatives(),
                        metrics.FalsePositives(),
                        metrics.TruePositives()])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 16)	5,120
lstm_1 (LSTM)	(None, 16)	2,112
dense (Dense)	(None, 1)	17

Total params: 14,500 (56.64 KB)
Trainable params: 7,249 (28.32 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 7,251 (28.33 KB)

Figure 23: Summary of LSTM model

The number for the first layer is given by:

$$Number\ of\ parameters = 4 * ((input\ size + bias\ term) * output\ size + output\ size^2)$$

$$\Rightarrow Number\ of\ parameters\ of\ first\ layer = 4 * ((63 + 1) * 16 + 16^2) = 5120$$

$$\Rightarrow Number\ of\ parameters\ of\ second\ layer = 4 * (16 + 1) * 16 + 16^2) = 2112$$

$$\Rightarrow Total\ number\ of\ parameters = 2 * (5120 + 2112 + 17) + 2 = 14500$$

4.2.2 Comparison of Different Dataset Splitting Ratio

For the LSTM model, we experimented with different training-to-validation dataset ratios, including 8:2 and 6:4. The comparison of these configurations is shown in the following table:

	8:2	6:4

Separation	`# Split data into training dataset and validation dataset`
	`# Divided into 8:2`
	
	`np.random.shuffle(all_data)`
	`split_pt = len(all_data) // 10 * 8`
	
	`train_data = all_data[:split_pt]`
	`valid_data = all_data[split_pt:]`
	
	`print("Number of Training Data:", len(train_data))`
	`print("Number of Validation Data:", len(valid_data))`
	
	`# Check if all data has been split`
	`print((len(train_data)+len(valid_data)) == len(label))`
	
	`Number of Training Data: 28216`
	`Number of Validation Data: 7063`
	`True`
		`# Split data into training dataset and validation dataset`
		`# Divided into 6:4`
		
		`np.random.shuffle(all_data)`
		`split_pt = len(all_data) // 10 * 6`
		
		`train_data = all_data[:split_pt]`
		`valid_data = all_data[split_pt:]`
		
		`print("Number of Training Data:", len(train_data))`
		`print("Number of Validation Data:", len(valid_data))`
		
		`# Check if all data has been split`
		`print((len(train_data)+len(valid_data)) == len(label))`
		
		`Number of Training Data: 21162`
		`Number of Validation Data: 14117`
		`True`

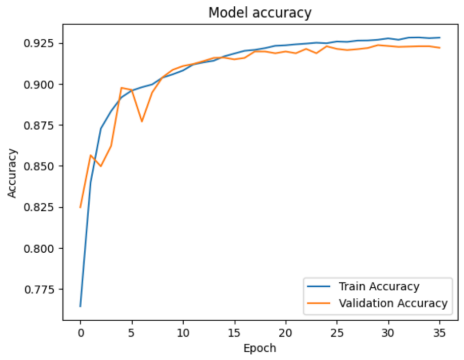
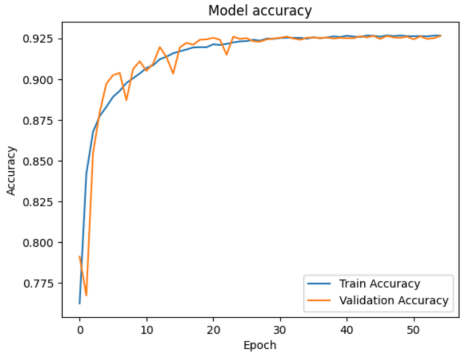
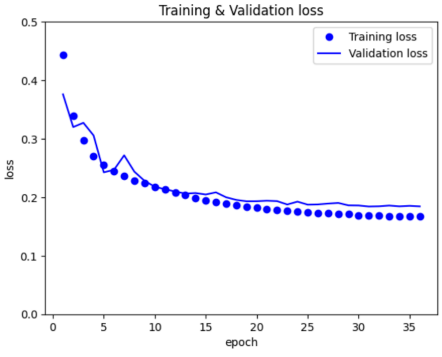
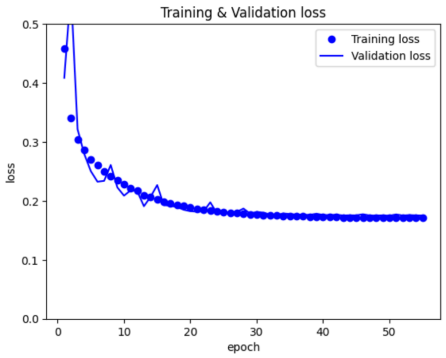
Accuracy		
Loss		
Training Accuracy	0.9283	0.9306

Table 2: Comparison of 8:2 & 6:4 splitting

In our comparison, we tried two different training-to-validation dataset splits: 80% for training and 20% for validation (8:2), and 60% for training and 40% for validation (6:4). This comparison was done to evaluate the impact of different dataset proportions on the model's performance, particularly in terms of overfitting and generalization.

Upon analyzing the accuracy and loss curves, we found that the 8:2 split showed a slight overfitting issue. Overfitting occurs when a model learns to perform well on the training data but struggles to generalize to new data. In our case, the accuracy on the training set was higher compared to the validation set, and this discrepancy became more evident as the training progressed. The loss curve for the 8:2 split indicated that the model's performance improved on the training set, while the validation loss plateaued or even worsened at times, confirming that the model was memorizing the training data instead of learning to generalize effectively.

When comparing the maximum accuracy achieved by each split, we observed that the 8:2 split reached a maximum accuracy of 0.9283, while the 6:4 split achieved a slightly better maximum accuracy of 0.9306. Although the difference in accuracy between the two splits is small, it is notable that the 6:4 split provided a slightly higher performance with no observable overfitting to the training data. This configuration appears to better balance training the model and evaluating its performance on unseen data.

The analysis indicates that while the 8:2 split allows the model to be trained on a larger portion of the data, it can lead to overfitting, as the model has fewer validation examples to test its generalization. On the other hand, the 6:4 split, despite having fewer training examples, helps the model better validate its performance during training, potentially leading to a better generalization capability. This suggests that the 6:4 split may be a more suitable choice when aiming for a model that can perform well on the training and new data.

In conclusion, while the 8:2 split provides a larger training dataset, which might be beneficial in some cases, the 6:4 split resulted in slightly better performance and reduced overfitting. The model trained with the 6:4 split showed higher accuracy.

4.3 YOLO Model

Figure 24: Example dataset of chair & bed

One of the significant challenges we are currently facing in our project is preparing a suitable dataset for YOLO model training. We found that existing datasets did not meet our specific requirements, particularly in terms of including hospital-specific objects such as beds and chairs. These objects are critical for our application, and the lack of a ready-made dataset necessitated the creation of a custom dataset tailored to our needs.

To address this, we began by capturing images of hospital beds and chairs during our internship in hospitals and recruiting from the Internet. These photos were then processed using Roboflow, where we manually annotated and labelled the objects in each image. The labelled images were subsequently converted into JSON/YAML format, which is compatible with YOLO for training purposes. To enhance the diversity of our dataset, we decided to integrate this custom dataset with an existing dataset containing generic objects like daily

life chairs and beds. By combining both datasets, we aim to create a more comprehensive dataset that can improve the accuracy of our YOLO model.

However, this approach has led to a second challenge: inconsistencies in the labelling schemes between our custom dataset and the existing dataset. Specifically, the labels we assigned to objects during our manual annotation process did not align with the naming conventions and structure used in the pre-existing dataset. For example, a "hospital bed" in our dataset might conflict with a "bed" label in the pre-existing dataset, causing issues when merging the two JSON/YAML files.

To resolve this, we are in the process of standardizing the labels across both datasets. This involves carefully examining the labelling conventions of both datasets, identifying mismatches, and updating labels to ensure consistency. It is a time-intensive process that requires attention to detail, as inconsistencies could lead to errors during model training or reduced performance of the YOLO model.

Both challenges—creating the custom dataset and resolving label inconsistencies—are still ongoing. Capturing images, annotating them, and converting them into the correct format requires significant effort. Similarly, merging and reconciling the datasets requires us to carefully ensure that the final, unified dataset is accurate and coherent. These steps are critical to building a high-quality dataset that will allow the YOLO model to perform effectively in recognizing hospital-specific objects.

CHAPTER 5: FUTURE DIRECTION

In the upcoming semester B, the focus will be on advancing the development of the AI LiDAR sensor for fall detection. The key objectives will include merging the YOLO object detection algorithm with LSTM networks to enhance the system's ability to analyze data for fall detection. This integration will allow for improved accuracy and sensitivity, and the model will undergo rigorous testing to calculate performance metrics such as accuracy and sensitivity. Through iterative modifications aimed at optimizing these parameters, a trained model with fixed weights can be developed.

To facilitate comprehensive testing, we will gather LiDAR sensor data to prepare for real-time applications to enhance processing speeds and responsiveness. Additionally, we will implement two operational modes to cater to different environments, such as a ward environment module and a backstair module, allowing for switching based on the setting.

A user-friendly graphical user interface (GUI) will be developed, featuring options to switch between modes, display the location of triggered alarms, and present soft alarm notifications. In terms of innovation, we will incorporate various soft alarms and implement a potential fall detection mechanism using LSTM to recognize specific scenarios, such as a partial leave from the bed. The interface will provide visual warnings, ensuring that caregivers or nurses are promptly alerted.

Furthermore, we aim to enhance the system's functionality by adding voice detection capabilities like a voice-activated safety bell. The soft alarm is triggered if a user calls out "Help", ensuring that caregivers can quickly respond. This feature will help prevent faults in the system, such as missed alarms, thereby improving the overall reliability and effectiveness of the fall detection system.

Through these advancements, we will create an effective and responsive AI LiDAR sensor system that significantly enhances the safety and well-being of the elderly.

Bibliography

[1] Department of Health. “Life Expectancy at Birth (Male and Female), 1971 – 2023.” Centre for Health Protection https://www.chp.gov.hk/en/statistics/data/10/27/111.html (accessed Dec. 12, 2024)

[2] L. Ren, & Y. Peng, "Research of Fall Detection and Fall Prevention Technologies: A Systematic Review," in IEEE Access, vol. 7, pp. 77702-77722, 2019, doi: 10.1109/ACCESS.2019.2922708.

[3] L. Yang, Y. Ren, and Zhang, W, “3D depth image analysis for indoor fall detection of elderly people.” Digital Communications and Networks, vol. 2, no. 1, pp. 24-34, Feb. 2016, doi: 10.1016/j.dcan.2015.12.001

[4] P. Vallabh and R. Malekian, “Fall detection monitoring systems: a comprehensive review.” Journal of Ambient Intelligence and Humanized Computing, vol. 9, pp. 1809–1833, Nov. 2018, doi: 10.1007/s12652-017-0592-3.

[5] S. Hu et al., "Radar-Based Fall Detection: A Survey [Survey]," in IEEE Robotics & Automation Magazine, vol. 31, no. 3, pp. 170-185, Sept. 2024, doi: 10.1109/MRA.2024.3352851.

[6] WHO. “Falls”. World Health Organization https://www.who.int/news-room/fact-sheets/detail/falls (accessed Dec. 12, 2024)

[7] W. Peng et al., “Demographics moderated the association of symptom burden with falls and fall-related outcomes”, Archives of Gerontology and Geriatrics, Vol. 117, 105190, Feb. 2024, doi: 10.1016/j.archger.2023.105190.

[8] O. Skorka and R. Ispasoiu, “Tradeoffs with RGB-IR image sensors,” *IEEE Transactions on Electron Devices*, vol. 69, no. 6, pp. 2915–2923, Jan. 2022, doi: 10.1109/ted.2021.3138381.

[9] D. Regazzoni, G. De Vecchi, and C. Rizzi, “RGB cams vs RGB-D sensors: Low cost motion capture technologies performances and limitations,” Journal of Manufacturing Systems, vol. 33, no. 4, pp. 719–728, Aug. 2014, doi: 10.1016/j.jmsy.2014.07.011.

[10] M. B. Shaikh and D. Chai, “RGB-D Data-Based Action Recognition: A Review,” *Sensors*, vol. 21, no. 12, p. 4246, Jun. 2021, doi: 10.3390/s21124246.

[11] M. Hasan et al., “LiDAR-based detection, tracking, and property estimation: A contemporary review,” Neurocomputing, vol. 506, pp. 393–405, Jul. 2022, doi: 10.1016/j.neucom.2022.07.087.

[12] A. Wu, W.-S. Zheng, S. Gong, and J. Lai, “RGB-IR Person Re-identification by Cross-Modality Similarity Preservation,” *International Journal of Computer Vision*, vol. 128, no. 6, pp. 1765–1785, Feb. 2020, doi: 10.1007/s11263-019-01290-1.

[13] J. Ma *et al.*, “A review of ToF-based LiDAR,” *Journal of Semiconductors*, vol. 45, no. 10, p. 101201, Oct. 2024, doi: 10.1088/1674-4926/24040015.

[14] N. Li et al., “A progress review on Solid‐State LiDAR and Nanophotonics‐Based LiDAR sensors,” Laser & Photonics Review, vol. 16, no. 11, Aug. 2022, doi: 10.1002/lpor.202100511.

[15] Shanghai ToFFuture Technology Co., Ltd, "XT-S240 240 Lines Solid-State Flash Lidar Product Manual," v1.3, Apr. 2024. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6241/XT-S240Mini.pdf. [Accessed: Dec. 16, 2024].

[16] A. Gunter, S. Boker, M. Konig, and M. Hoffmann, “Privacy-preserving People Detection Enabled by Solid State LiDAR,” 2020 16th International Conference on Intelligent Environments (IE), Jan. 2020, doi: 10.1109/ie49459.2020.9154970.

[17] Sick AG, “TIM-Series 2D LIDAR sensors,” 8014314, Dec. 2020. [Online]. Available: https://docs.rs-online.com/bb19/A700000010029400.pdf. [Accessed: Dec. 16, 2024].

[18] Intel®, “Intel® RealSenseTM LiDAR Camera L515 Datasheet,” Revision 003, Jan. 2021. [Online]. Available: https://www.mouser.com/datasheet/2/612/Intel_RealSense_LiDAR_L515_Datasheet_Rev002-171

3847.pdf?srsltid=AfmBOoqSHbLyqEt-dX3bNaLUVYgOoGfYpq3oaR2SV8LoCLBZ24j694LB. [Accessed: Dec. 16, 2024].

[19] Unitree Robotics, “Unitree 4D LiDAR-L1 User Manual,” v1.1, Jun. 2024. [Online]. Available: https://unitree-website.oss-cn-hangzhou.aliyuncs.com/Lidar/L1%20Quike%20Start%20Guide_v1.0.pdf. [Accessed: Dec. 16, 2024].

[20] N. T. Newaz and E. Hanada, “The Methods of Fall Detection: A Literature review,” *Sensors*, vol. 23, no. 11, p. 5212, May 2023, doi: 10.3390/s23115212.

[21] F. Harrou, N. Zerrouki, Y. Sun, and A. Houacine, “A simple strategy for fall events detection,” 2022 IEEE 20th International Conference on Industrial Informatics (INDIN), vol. 31, pp. 332–336, Jul. 2016, doi: 10.1109/indin.2016.7819182.

[22] N. Zerrouki and A. Houacine, “Combined curvelets and hidden Markov models for human fall detection,” Multimedia Tools and Applications, vol. 77, no. 5, pp. 6405–6424, Mar. 2017, doi: 10.1007/s11042-017-4549-5.

[23] M. Paul, S. M. E. Haque, and S. Chakraborty, “Human detection in surveillance videos and its applications - a review,” EURASIP Journal on Advances in Signal Processing, vol. 2013, no. 1, Nov. 2013, doi: 10.1186/1687-6180-2013-176.

[24] N. L. Wang, N. X. Geng, C. Leckie, and R. Kotagiri, “Moving shape dynamics: A signal processing perspective,” 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, Jun. 2008, doi: 10.1109/cvpr.2008.4587554.

[25] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Robust video surveillance for fall detection based on human shape deformation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 5, pp. 611–622, Mar. 2011, doi: 10.1109/tcsvt.2011.2129370.

[26] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Fall Detection from Human Shape and Motion History Using Video Surveillance,” 21st International Conference on Advanced

Information Networking and Applications Workshops (AINAW'07), Jan. 2007, doi: 10.1109/ainaw.2007.181.

[27] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *arXiv (Cornell University)*, Jan. 2016, doi: 10.48550/arxiv.1611.08050.

[28] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D pose Estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, Jul. 2019, doi: 10.1109/tpami.2019.2929257.

[29] S. Kreiss, L. Bertoni, and A. Alahi, "OpenPIFPAF: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13498–13511, Nov. 2021, doi: 10.1109/tits.2021.3124981.

[30] W. Chen, Z. Jiang, H. Guo, and X. Ni, "Fall detection based on key points of Human-Skeleton using OpenPose," *Symmetry*, vol. 12, no. 5, p. 744, May 2020, doi: 10.3390/sym12050744.

[31] Workshop, "Fall Dataset," Roboflow Universe, 2024. [Online]. Available: https://universe.roboflow.com/workshop-yg2yt/fall-dtk98. [Accessed: Dec. 16, 2024].

[32] N. D. Dao, T. V. Le, H. T. M. Tran, Y. T. H. Nguyen, and T. D. Duy, "The Combination of Face Identification and Action Recognition for Fall Detection," *The University of Danang - Journal of Science and Technology*, vol. 20, no. 12.2, pp. 37–44, 2022, doi: 10.31130/ud-jst.2022.539ict.

[33] H. Zhang *et al.*, "Deep learning-based 3D point cloud classification: A systematic survey and outlook," *Displays*, vol. 79, p. 102456, May 2023, doi: 10.1016/j.displa.2023.102456.

[34] E. Auvinet, L. Reveret, A. St-Arnaud, J. Rousseau, and J. Meunier, "Fall detection using multiple cameras," 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 2554–2557, Aug. 2008, doi: 10.1109/iembs.2008.4649721.

[35] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," *Computer Methods and Programs in Biomedicine*, vol. 117, no. 3, pp. 489–501, 2014. doi: 10.1016/j.cmpb.2014.09.005.

[36] G. Hidalgo et al., "OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation." GitHub, https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[37] R. V. Iyer, P. S. Ringe, and K. P. Bhensdadiya, “Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 8, no. 7, pp. 1156–1170, Jul. 2021, [Online]. Available: https://www.researchgate.net/profile/Rakkshab-Iyer/publication/353211011_Comparison_of_YOLOv3_YOLOv5s_and_MobileNet-SSD_V2_for_Real-Time_Mask_Detection/links/60ed405c0859317dbddb6891/Comparison-of-YOLOv3-YOLOv5s-and-MobileNet-SSD-V2-for-Real-Time-Mask-Detection.pdf

[38] M. Hussain, “YOLOV5, YOLOV8 and YOLOV10: the Go-To detectors for real-time vision,” *arXiv (Cornell University)*, Jul. 2024, doi: 10.48550/arxiv.2407.02988.

[39] F. Xie, B. Lin, and Y. Liu, “Research on the coordinate attention Mechanism fuse in a YOLOV5 deep learning detector for the SAR ship detection task,” *Sensors*, vol. 22, no. 9, p. 3370, Apr. 2022, doi: 10.3390/s22093370.

[40] S. Nikhileswara Rao, "YOLOv11 architecture explained: Next-level object detection with enhanced speed and accuracy," Medium, 2024. https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71.

[41]M. Sohan, T. S. Ram, and Ch. V. R. Reddy, “A review on YOLOV8 and its advancements,” *Algorithms for Intelligent Systems*, pp. 529–545, Jan. 2024, doi: 10.1007/978-981-99-7962-2_39.

[42] R. Khanam and M. Hussain, “YOLOV11: An overview of the key architectural enhancements,” *arXiv (Cornell University)*, Oct. 2024, doi: 10.48550/arxiv.2410.17725.

[43] Sawant, C. (2020). "Human activity recognition with OpenPose and Long Short-Term Memory on real-time images."[Online]. Available: https://api.semanticscholar.org/CorpusID:210969352

[44] F.-E. Ait-Bennacer, A. Aaroud, K. Akodadi, and B. Cherradi, “Applying Deep Learning and Computer Vision Techniques for an e-Sport and Smart Coaching System Using a Multiview Dataset: Case of Shotokan Karate,” *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 18, no. 12, pp. 35–53, Sep. 2022, doi: 10.3991/ijoe.v18i12.30893.

[45] D. Kumar and A. Sinha, “Yoga pose detection and classification using deep learning,” *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, pp. 160–184, Nov. 2020, doi: 10.32628/cseit206623.

[46] E. Alam, A. Sufian, P. Dutta, and M. Leo, “Vision-based human fall detection systems using deep learning: A review,” *Computers in Biology and Medicine*, vol. 146, p. 105626, May 2022, doi: 10.1016/j.compbiomed.2022.105626.

[47] M. M. Hasan, M. S. Islam, and S. Abdullah, “Robust Pose-Based Human Fall Detection Using Recurrent Neural Network,” *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, pp. 48–51, Nov. 2019, doi: 10.1109/raaicon48939.2019.23.

[48] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 5929–5955, 2020. doi: 10.1007/s10462-020-09838-1.

[49] S. S. Jeong, N. H. Kim, and Y. S. Yu, "Fall detection system based on simple threshold method and long short-term memory: Comparison with hidden Markov model and extraction of optimal parameters," *Applied Sciences*, vol. 12, no. 21, p. 11031, 2022. doi: 10.3390/app122111031.